

AD-A124 206

EDITSPEC: SYSTEM MANUAL VOLUME II SYSTEM DESIGN  
CONCEPTS REVISION(U) CONSTRUCTION ENGINEERING RESEARCH  
LAB (ARMY) CHAMPAIGN IL E S NEELY NOV 79

1/1

UNCLASSIFIED

DOD/DF-83/002D

F/G 9/2

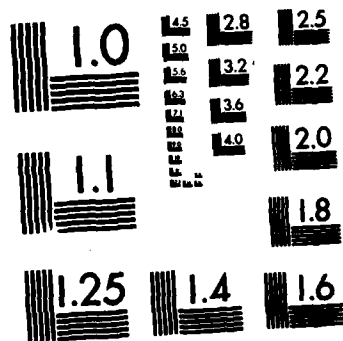
NL


END

FILED

11

DTIC



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

ADA 124206

0

EDITSPEC SYSTEM MANUAL  
VOLUME II: SYSTEM DESIGN CONCEPTS

by  
E. S. Neely

January 1978  
revised November 1979

DTIC  
ELECTE  
FEB 8 1983  
A

DTIC FILE COPY

REPRODUCED BY  
NATIONAL TECHNICAL  
INFORMATION SERVICE  
U.S. DEPARTMENT OF COMMERCE  
SPRINGFIELD, VA. 22161

83 02 08 138

This document has been approved  
for public release and sale; its  
distribution is unlimited.

<b>REPORT DOCUMENTATION PAGE</b>	<b>1. REPORT NO.</b> DOD/DF-83/002d	<b>2.</b>	<b>3. Recipient's Accession No.</b> AD A124 206
<b>4. Title and Subtitle</b> EDITSPEC: System Manual, Volume II: System Design Concepts			<b>5. Report Date</b> November 1979
<b>7. Author(s)</b> Edgar S. Neely, Jr.			<b>6.</b>
<b>9. Performing Organization Name and Address</b> Department of the Army Construction Engineering Research Laboratory P.O. Box 4005 Champaign, IL 61820			<b>8. Performing Organization Rept. No.</b>
<b>12. Sponsoring Organization Name and Address</b> (same)			<b>10. Project/Task/Work Unit No.</b> 4A762731AT41/T1/009
			<b>11. Contract(C) or Grant(G) No.</b> (C) (G)
			<b>13. Type of Report &amp; Period Covered</b>
			<b>14.</b>
<b>15. Supplementary Notes</b>  For magnetic tapes, see			
<b>16. Abstract (Limit: 200 words)</b> The EDITSPEC System is an automated system designed to produce construction specifications from Corps of Engineers Guide Specifications. The System uses one central computer and a communications network to provide remote terminal access by Corps offices, nationwide to a central data base.  This volume provides computer system personnel with the basic system design concepts used in developing EDITSPEC. The relationship between the design concepts and the data and common storage structures within EDITSPEC is also indicated. A description of the basic functions of the system is given in the remaining chapters of the report.			
<b>17. Document Analysis a. Descriptors</b> Construction Specifications Guide Specifications Military Construction  <b>b. Identifiers/Open-Ended Terms</b>    <b>c. COSATI Field/Group</b>			
<b>18. Availability Statement:</b>		<b>19. Security Class (This Report)</b> UNCLASSIFIED	<b>21. No. of Pages</b>
		<b>20. Security Class (This Page)</b> UNCLASSIFIED	<b>22. Price</b>

## PREFACE

The purpose of the EDITSPEC system reports is to provide complete documentation to all personnel involved with the EDITSPEC system. Such personnel include managers, specification writers, typists, computer systems analysts, and computer programmers. Each personnel group requires different documentation. The reports required for each group and the order in which they should be read are shown in the table below.



DTIC	
COPIES	
INSPECTED	
2	
DTIC GRAFI	
DTIC TAB	
Unannounced	
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

PERSONNEL	MANA-	SPEC	SYSTEM	PRO-
REPORTS	GERS	WRITERS	ANALYST	GRAMMER
1. Const. Spec. Prep				
within EDITSPEC system	1	1	1	1
2. Preparing a Guide				
Spec for EDITSPEC		4		
3. TRAINING COURSE				
MANUALS				
a. instructor				
b. instroduction	2	2	2	2
c. specwriter		3	3	
d. internal				
commands		3		
e. Edit				
commands		4		
f. system				
commands		5		
4. USER'S MANUAL		Reference	2	2
5. SYSTEM OVER-				
VIEW			3	3
6. SYSTEM DESIGN				
CONCEPTS			4	4

7. TABLE HANDLER		5	5
8. DATA HANDLER		6	6
9. CONVERSION and EXTENSION	2	7	
10. Project Manager's Procedures		8	
11. Operation & Maintenance Procedures		9	

## ABSTRACT

This volume provides computer system personnel with the basic system design concepts used in developing EDITSPEC, a computer-aided system for preparing construction project specifications. The relationship between the design concepts and the data and common storage structures within EDITSPEC is also indicated.

→ The basic functions of EDITSPEC have been grouped together in the remaining chapters of this report. Chapter 1 addresses the resource management functions. Chapters 2 through 5 discuss functions related to the text itself. Chapter 6 describes the edit history kept by the system. Chapter 7 covers the listing information stored within EDITSPEC. Chapters 8 through 10 deal with information on formatting and printing. Command execution procedures are given in Chapter 11. Chapter 12 discusses the data-base protection functions. Chapter 13 addresses the program and data conversion functions.

↑

## FOREWORD

This investigation was performed by the Directorate of Military Construction. Office of the Chief of Engineers (OCE), under Project 4A762731AT41, "Design, Construction, and Operation and Maintenance Technology for Military Facilities"; Task T1, "Development of Automated Procedures for Military Construction"; Work Unit 009, "Computer-Based Specifications." The applicable QCR is 1.10.001. The OCE Technical Monitor was William Darnell.

The study was performed by the Management Systems Branch (Dr. O. E. Rood, Jr., Chief), Facility Acquisition and Construction Division (Mr. E. A. Lotz, Chief), U.S. Army Construction Engineering Research Laboratory (CERL).

COL J. E. Hays is Commander and Director of CERL and Dr. L. R. Shaffer is Technical Director.

### List of Figures

- 1 User Information Structure
- 2 Document Table Structure
- 3 Typical Specification Condition List
- 4 Specification Condition Structure
- 5 Header Data Structure
- 6 Footer Data Structure
- 7 Typical Paragraph Format and Data Structure
- 8 System Design - Paragraph Numbering Format
- 9 System Design - Page Numbering Format
- 10 Vertical Page Layout
- 11 Horizontal Page Layout
- 12 Paragraph Form Structure
- 13 Page Form Structure
- 14 PGBUF Structure

## 1 RESOURCE MANAGEMENT

### Introduction

Within the EDITSPEC system, three types of resources require close management and security. The first type is the group of people that will use the system. The second type is the collection of account numbers that will be used in the system. The third is the collection of text documents that will be entered into the system. These resources are discussed in the following sections on personnel, financial, and document management, respectively.

### Personnel Management and Security

#### *Objective*

In the EDITSPEC system, management personnel must be able to control the access of other individuals to the system. The current tree-structure personnel management scheme must work within the EDITSPEC system in the same manner as in the current manual system for preparing specifications. Each person must have control over all of his resources including his system access code. Each individual's supervisor has control of the individual's access to the system. Personnel should be able to communicate within the system in the same manner as they now do by mail.

#### *Data Storage Structure*

The data are stored in the System User Table as shown in Figure 1. The user's password, located in data record type 1, allows each person to control access to the system by other personnel who know the user's

identification (ID) code. If a user is a supervisor of other users, the supervisor switch is turned on. When the user has access to more than 5 account numbers and/or more than 25 documents, the lists are continued in type 2 and type 3 data records, respectively. If messages have been sent to the user, the first message is pointed to in the type 1 record, and all such messages are stored in type 4 records. The user ID code of the supervisor of this user is recorded. The first 5 account numbers and the first 25 document names accessible by this user are listed at the close of the type 1 data record.

#### *Common Storage Structure*

The user ID code is stored in the array USID of COMMON/.SYSTEM/. The supervisor switch is stored in ISUP in the same command.

#### *Related Commands*

- .LOGON
- .LOGOF
- .USER
- .CANC
- .SUPE
- .MESS
- .ACCE
- .RACC

## Financial Management and Security

### *Objective*

The system must be able to control the access of personnel to the account charge numbers within the system. The current tree-structure account access management scheme is different from the personnel-management scheme, and must work within the EDITSPEC system. The same type of accounting report produced under the current system must also be produced by EDITSPEC.

### *Data Storage Structure*

The ID code of the users and supervisors associated with each account number are stored in the system Account Number Table.

### *Common Storage Structure*

The account number is stored in the COMMON/LOGIC/ in array ACTNO.

### *Related Commands*

.ACCO

.REPO

.ACCE

.RACC

## Document Management and Security

### *Objective*

The person who creates a particular specification document must have complete control over the access of the other personnel to his document and the type of access that each person will have.

### *Data Storage Structure*

The basic unit for text storage is the document. Each document is identified by a unique document name and is composed of tables, as shown in Figure 2. The purpose of each table is described below.

Text Table. All text related to the document is stored within the text table in an unformatted mode by line number. A text-table line, which is not the same as a printed line, may contain up to 480 characters. (The length of a printed line is controlled by the user and specified only when a print request is given.) All text characters are stored in the text table in EDITSPEC character notation. That notation is the same as for typewritten characters with two exceptions: (1) every capital letter must be represented in EDITSPEC by two characters--the cent sign and the corresponding lower case letter (e.g., the phrase "Go to CERL" would be stored in EDITSPEC as: cgo to cccecrcl); (2) a string of asterisks to be printed is entered as the desired asterisk string plus one additional asterisk (e.g., one asterisk to be printed is entered as two asterisks; four asterisks to be printed are entered as five).

Each text line can contain a maximum of 480 EDITSPEC characters. Each document can contain a maximum of 99,999,999 lines, which is equivalent to approximately 12,000,000 pages of single-spaced text.

Flag Table. The text often contains sections where the user must choose between several small phrases. Such sections are marked with a flag within the text. The phrase options and phrase selected are defined in the flag table.

Pull Table. The EDITSPEC system can create a new document from an existing document by pulling text lines from the old document and placing them into the new document. The instructions for doing so are stored in the pull table.

Logic Condition Table. The description of an item is often dispersed throughout several documents. To insure that the complete definition of the item will be contained in the documents and to insure that the line containing the definition will not be inadvertently removed, the text may be logically connected by listing the appropriate document names and line numbers. Such logical connections are stored in the logic condition table.

Contents Table. Most documents require several types of contents tables, such as the standard table of contents, a list of tables, and a list of figures. The entries for all types are stored in the contents table.

Index Table. Documents often require different types of index tables. The entries for all types are stored in the index table.

External Reference Table. When the writer plans for one document to be created by copying text from a second document during printing, or when the creator of one document has defined a logic condition involving a second document, the name of the first document and the type and number of applications are stored within the appropriate external reference table of second document. The latter document cannot be deleted if there are any external references within these tables.

Audit--General Table. This table contains a record of all editing performed on the document, including the user's identification, date, and time.

Audit--Text Table. A complete history of all changes made to the text of the document can be maintained in this table.

Audit--Other Table. A complete history of all changes made to the other information within the document can be maintained in this table.

Document Reference Table. This table contains a list of all documents referenced by a user-entered copy command.

#### *Command Storage Structure*

All basic information related to the document is located in the /EDITC/ common block.

#### *Related Commands*

.NEW

.EDIT

.STOR

.RENA

.DELE

.ACCE

.RACC

## 2 TEXT INPUT

### Text Input From Input Devices

#### *Objective*

The system should allow the user to enter small phrases of text as well as bulk text consisting of many lines from any terminal or input device. The user should have complete control over the positioning of the text entered within the line.

#### *Data Storage Structure*

The data are stored in the Text Table by line numbers :

#### *Common Storage Structure*

No common areas are required.

#### *Related Commands*

.IN

.EN

.AB

.IT

.AA

### Text Input From Documents

#### *Objective*

The system should allow the user to enter text from documents stored in the system. The user should control when the text will be entered.

Text should be entered either immediately after the command is given or not until the printing of a document.

*Data Storage Structure*

The data are stored in the Text Table by line number.

*Common Storage Structure*

No common areas are required.

*Related Commands*

.CO	*CO*
.CT	*CT*
.CR	*CU*
.MO	*CD*

### 3 TEXT CORRECTION

#### Objective

The system must be able to locate, change, and erase text as described in this section. The user should have the option of specifying that the text should be processed exactly as entered or according to the following set of rules. The text to be located should be found regardless of whether it is given in lowercase, uppercase, or mixed lowercase and uppercase letters. The text to be replaced should be placed according to the case of the current text. If the current text is all in uppercase letters, the replacement should be made in all uppercase letters. If the current text is all lowercase, the replacement should be all lowercase. When the first letter of the correct phrase is capitalized, the first character of the replacement text should be capitalized.

The system should process all English punctuation correctly. If the word *GREEN* is to be replaced by *RED* in the sentence "The color is *GREEN*," the new sentence should read "The color is *RED*."

The user should have the capability of requesting that the system report all near matches for the text. A near match is defined as a phrase that would not have the same number of leading or trailing blanks.

#### Data Storage Structure

Data are stored within the Text Table.

### Common Storage Structure

No common storage is required.

### Related Commands

.LO

.CH

.ER

## 4 TEXT IDENTIFICATION AND CLASSIFICATION

### Text Identification

#### *Objective*

The text should be identified by line numbers for fast access and editing. The line numbers should not change during normal editing, since any changes would confuse the user during future editing sessions. The increment between two consecutive line numbers is under the control of the user, and the user can renumber the lines.

#### *Data Storage Structure*

The text is stored within the document Text Table according to line numbers.

#### *Common Storage Structure*

No common area is required.

#### *Related Commands*

.LI

.RN.

All input commands

### Text Classification

#### *Objective*

The user should have the ability to classify text according to who will use the text. For example, sections of the text could be classified as contractual text, notes to the specification writer, or notes for the estimator.

#### *Data Storage Structure*

The text classifications are known as text segments and are stored with the text in the Text Table.

#### *Common Storage Structure*

No common area is required.

#### *Related Commands*

.TS.

All input commands.

## 5 AUTOMATIC CREATION OF PROJECT SPECIFICATIONS

### Introduction

The EDITSPEC system can obtain a required segment of text from the guide specification and place it into the project specification being created. EDITSPEC must be able to update the project specification when changes occur either in the design or in the guide specification. The system automatically selects the correct phrases according to the project design and insures that all necessary text referring to an item in the design is included within the project specification.

The system insures that the latest version of the guide specification is always incorporated into the project construction specification, which is part of the construction bid package. As soon as the bid package is completed, all references to other documents are removed from the project specification.

### Project Design

#### *Objective*

The specification writers must have a method for defining the specification conditions for the projects under design. The computer system must be capable of checking the specification conditions for the project to insure that they are complete. Security must be maintained so that only the project specification writer is allowed to add, delete, or change the specification condition values.

### *Data Storage Structure*

The specification conditions are defined for each guide specification and numbered sequentially, starting from number one. An example of a specification list is given in Figure 3.<sup>25</sup> The written definitions are not stored within the computer system. Instead, two bits are stored for each specification condition for each project. These bits indicate the current status of the condition--whether it has been selected or rejected, or whether no decision has been made.

The specification condition values are stored in the system Specification Condition Table in the structure shown in Figure 4.

### *Common Storage Structure*

No common storage required.

### *Related Commands*

.SPEC

## Obtaining Text From Guide Specifications

### *Objective*

The guide specification writer must have a method for instructing the computer system how to "pull" text from the guide specification and to place the text into the project specification. The system must allow any specification to serve as the basis for the specification being created; project specifications can be created not only from guide specifications but also from previously created project specifications.

### *Data Storage Structure*

The pull command data are stored in the documents Pull Table. The pull commands are stored in the order in which they must be executed. Each pull command contains a list of specification conditions, an indicator of whether one or all of the specification conditions must be met before the command is executed, and the procedure to be executed if the specification conditions are met.

### *Common Storage Structure*

During the execution of the generation and updating programs, the pull commands are stored in the /GEDAC/ common area.

### *Related Commands*

.PU

## Automatic Phrase Selection

### *Objective*

In certain parts of the guide specifications, the project specification writer would normally have to select one of several phrases, as determined by the project specification conditions. The computer system should be able to perform the selection automatically.

### *Data Storage Structure*

The location in the guide specification where the writer is offered a multiple-choice selection of phrases is known as a flag. The allowable phrases and the current phrase to be applied are stored in the document Flag Table.

#### *Common Storage Structure*

No common storage is required.

#### *Related Commands*

.FL

.SC      \*FL\*

.FC

.PU

#### Complete Design Item Definition

#### *Objective*

The system must have a method by which the guide and project specification writers can define all of the text required to completely describe a construction item. The system must also insure that all text is present before a specification is printed.

#### *Data Storage Structure*

The data related to the logical description of an item are stored within the documents Logic Condition Table. The name of each document containing text related to the item and the actual line numbers of the appropriate text are stored in the logic condition command.

#### *Common Storage Structure*

No common storage is required.

#### *Related Commands*

.RL

## Generating and Updating Project Specifications

### *Objective*

The system must be able to generate a project specification from a guide specification when given the actual project design. The system must be able to update this project specification after changes to either the guide specification or the project design.

### *Data Storage Structure*

The Specification Condition and Pull Common Tables are used to perform generating and updating functions.

### *Common Storage Structure*

The specification condition values and the pull commands are stored within the /GEDAC/ common area.

### *Related Commands*

.GENE	.CF	*CO
.UPDA	.LC	*CD
.UPME		*CT
		*CU

## Removing External References

### *Objective*

When the project specification is ready for the printing of the bid package, all references to other documents should be resolved and the referenced text incorporated into the project specification. Amendments and change orders will be processed against this new project specification.

*Data Storage Structure*

All document tables are applied.

*Common Storage Structure*

No common storage is required.

*Related Commands*

.RR

NOTES ON THE USE OF  
OCE GUIDE SPECIFICATIONS

A. The following specification conditions should be evaluated when a project requires strip shingles. (Reference CE 220.02 ROOFING; STRIP SHINGLES)

1. This project contains strip shingles.
2. This project has a roof slope between 3.00 and 3.99 in./ft.
3. This project is located in a region having a normal low temperature below -20°F.
4. This project is located in a region having prevalent winds greater than 40 mph.
5. This project has valleys.
6. This project is located in a region having prevalent winds less than 40 mph.
7. This project has a roof slope greater than 4 in./ft.
8. This project is located in a region having a normal low temperature above -20°F.
9. This project contains either hips or ridges.

Figure 3. Typical specification condition list.

## 6 DOCUMENT EDITING HISTORY

The system keeps a complete audit history of all changes made to the document and uses this history to recover previous versions of the text.

### General Audit Information

#### *Objective*

The system always keeps a record of each person that has changed the document. The date and time of the change and the reason for the change should be recorded.

#### *Data Storage Structure*

The data are stored by edit cycle number in the Audit Trail--General Table.

#### *Common Storage Structure*

No common storage is required.

#### *Related Commands*

.LT

.AU

### Text Audit Information

#### *Objective*

The system should allow the user to keep a complete audit record of every change made to each text line. This audit trail takes the place of the several marked-up hard copies that are saved when specifications are prepared manually.

#### *Data Storage Structure*

The data are stored by line number and edit cycle number in the Audit Trail--Text Table.

Each line of text is identified as a logical unit. When a new line of text is added to a document it becomes a new logical unit. When an existing text line is changed, the entire changed line is entered into the Audit Trail Table at the end of the table and pointed to in the header. The cycle numbers are stored in sequential order and are referenced by an Audit Table--General Pointer.

The current line numbers are stored in an ordered key array for fast access to the logical test.

When the line numbers of the document are resequenced the header records containing the current line numbers are resequenced also. The old and new line numbers are stored intact in the general audit trail table immediately after the general information record.

#### Table Audit Information

The system allows the user to keep a complete audit record of each change made to each document table. As with the text audit trail, the table replaces the several marked up hard copies that would normally be saved.

#### *Data Storage Structure*

The data are stored by table name and edit cycle number in the Audit Trail--Table Table.

#### *Common Storage Structure*

No common storage is required.

#### *Related Commands*

.AU

.LT

#### Removing and Recreating Edit Cycles

##### *Objective*

The system should allow the user to remove unwanted information on older edit cycles and to recreate the text as it was during a previous cycle.

##### *Data Storage Structure*

The data are stored in the Audit Trail--Text Table.

#### *Common Storage Structure*

No common storage is required.

#### *Related Commands*

.MC

.RC

## 7 LISTING INFORMATION

The EDITSPEC system provides the user with an easy method for obtaining information stored within the system.

### Listing System Information

#### *Objective*

The system must provide the user with an easy method for obtaining such items of information as valid user IDs, account numbers, document names, and standard formats.

#### *Data Storage Structure*

The data are stored in a different structure for each table.

#### *Common Storage Structure*

The structure of the common storage is also different for each table. Each listing is produced in a free-formatted mode.

#### *Related Commands*

.LIST

### Listing Document Information

#### *Objective*

The system must provide the user with an easy method for obtaining information such as flag definitions, text, and external references.

#### *Data Storage Structure*

The structure used for data storage is different for each table.

#### *Common Storage Structure*

The structure of the common storage is also different for each table.

Each listing is produced in a free-formatted mode.

#### *Related Commands*

.LT

## 8 DEFINING SYSTEM PRINTING FORMATS

The system provides functions for defining standard formats for printing headers, footers, paragraphs, paragraph numbers, page numbers, and document printing formats. These formats can be applied during print operations.

### Header Formatting

#### *Objective*

The system must provide a function for defining the header format but also for defining different formats for even and odd pages, as shown in Figure 5. The total length of the header would not exceed eight text lines. A method for changing a portion of the header at the time the text is printed must be provided for use in preparing family housing specifications.

#### *Data Storage Structure*

The headers are stored as entered in the header format table.

#### *Common Storage Structure*

The heading format definitions are applied during the printing of a document. The system stores the header for the odd pages in /PAGEBF/ common in array IHEAD0. The header for the even pages are stored in array IHEAD1. The number of lines in each header is stored as variables HDLNO and HDLNE, respectively. The locations of the header line containing the changeable header text are variables IHEAO and IHEAE, respectively. The heading text to be placed in the changeable portion of the header is

stored in array IHEADS. If this text has been changed during the typing of a page, variable IHC is set to one to indicate the required change.

#### *Related Commands*

.HEAD      \*HS

.DOCU

#### Footer Formatting

##### *Objective*

The system must provide a function for defining the format of footers in exactly the same manner as for headers. The capability for defining different formats for even and odd pages is again provided, as shown in Figure 6. The total length of the footer does not exceed eight text lines. A method for changing a portion of the footer at the time the text is printed must be provided for preparing family housing specifications.

##### *Data Storage Structure*

The footers are stored as entered in the footer format table.

##### *Common Storage Structure*

The footer format definitions are applied during the printing of a document. The system stores the footer for the odd pages in /PAGEBF/ common in array IF00T0. The footer for the even pages are stored in array IF00TE. The number of lines in each footer is stored in variables FILNO and FTLNE, respectively. The locations of the footer line containing the changeable footer text are variable IF000 and IF00E, respectively. The footer text

to be placed in the changeable portion of the footer is stored in array IFFOOTS. If this text has been changed during the typing of a page, variable IFC is set to one to indicate the required change.

#### *Related Commands*

.FOOT      \*FS

.DOCU

#### Paragraph Formatting

##### *Objective*

The system must provide a function for defining the print format to be applied in printing a paragraph. A typical paragraph is shown in Figure 7, which indicates the items that must be defined. The left and right page margins are included for reference. The paragraph format must be identified so that it can be referenced within the text of a document. The first item that the typist must know is the number of spaces from the left margin that the first character of the first line in the paragraph is to be indented. The next item is the number of spaces to place after the paragraph number and before the first word in the paragraph. Two spaces will always be automatically placed after the paragraph number. The indentation from the left margin for the first character of the second through the last line must also be defined, as must the indentation from the right margin for the last character in all lines.

The paragraph format ID and the three indentions will comprise the paragraph format definition.

One blank line will be skipped after each paragraph.

### *Common Storage Structure*

The paragraph format definitions are applied during the printing of the document text. Most text will require the use of several paragraph formats. The computer system can store up to 12 paragraph format definitions in the PAGEBF common area in array IPFT (see Figure 7). If a 13th paragraph format is entered, it will overwrite (take the place of) the 12th format. The number of active formats in IPFT is given in NPFC. The most commonly used paragraph formats are defined in the document format table. These paragraph formats are automatically loaded into the common area during the printing initialization process.

### *Data Storage Structure*

The data items are stored within the paragraph format table as shown in Figure 7.

### *Related Commands*

.PARA      \*P

.DOCU

### Paragraph Number Formatting

#### *Objective*

The system must provide a function which defines the print format to be applied in printing a particular paragraph number. A typical paragraph number, 12.AB.(12)(ab)(IV).(iv).., is shown in Figure 8. The items that must be defined are also shown in the figure. The paragraph number is composed of subfields that relate to levels of detail or indentation of the

paragraphs. No more than six levels of indentation or subfields should be required for processing the text.

Each subfield may be composed of leading and/or trailing special characters that are not incremented but indicate the level of indentation. The leading and trailing special characters are optional and, if present, can be of any length. The total length of the number should not exceed 40 characters. The part of each subfield to be incremented may consist of an arabic number, uppercase or lowercase roman numerals, or alphabetic characters. The length of each subfield increment may vary.

Two blank spaces should always be left after the paragraph number.

#### *Data Storage Structure*

The data items stored within the paragraph numbering format table are shown in Figure 8 as the FORMAT DATA RECORD. The first four sets of entries have values for each of the six possible subfields. The first set of entries, labeled TYPE N, contains a numerical code for the basic type of subfield increment. A minus one is used to indicate an alphabetic increment, a zero indicates an arabic number, and a positive number indicates a roman numeral. The positive number indicates the arabic representation of the next roman-numeral paragraph number to be assigned.

The second entry set, labeled TYPE A, contains an alphabetic code for the exact type of subfield increment applied. The letter b is used to indicate arabic numbering, c indicates uppercase alphabetic, a indicates lowercase alphabetic, t designates uppercase roman, and r indicates lowercase roman subfield increments. The third set contains the character

location for the last character to be incremented in the subfield. The fourth set contains the character location for the last character to be printed in the subfield.

The next two words contain the total number of characters in the paragraph number followed by the actual number of subfields defined by the user. The last item of data in the record is the initialized value of the paragraph number (one character per word). The paragraph number is initialized to zero for arabic numbers and to blanks for all other increment types.

#### *Common Storage Structure*

The paragraph number format definitions are applied during the printing of a document text. Most text will require the use of several paragraph number formats. The computer system will store the current paragraph number format definitions in the NUMBRC common area in array IPNFC. The initial values for each of the six formats are stored in array INPNFC. The variable IPNFCC contains the number of paragraph number formats actually used.

When the seventh paragraph numbering format is requested, it will overwrite the sixth format.

The first set of six words in the common array contains the number of characters within the incrementable portion of the subfield. The next word contains the paragraph numbering format ID. The next two words are not currently applied. The remaining common area is used as described in the previous section.

### *Related Commands*

.PRNF	*P	.BP
.DOCU	*BP	

### Page Number Formatting

The system allows the user to define the print format to be applied in printing a page number. A typical page number and the items that must be defined are shown in Figure 9. The page number is composed of subfields that relate to levels of detail of the document. No more than six levels of detail or subfields can be required for processing the text.

Each subfield may be composed of leading and/or trailing special characters that are not incremented but indicate the level of detail. The leading and trailing special characters are optional and, if present, can be of any length. The total length of the number should not exceed 40 characters. The part of each subfield to be incremented may consist of arabic numbers, uppercase or lowercase alphabetic characters, or uppercase or lowercase roman numerals. The length of each subfield increment may vary.

The page number may appear at the bottom or top of a page. The number may be located at the left, center, or right side of each page. The page number may also be located on the outside edge of each printed page. The printing of the page number may begin on the first, second, or third page of any document.

### *Data Storage Structure*

The data items stored within the page numbering format table are shown in Figure 9 as the FORMAT DATA RECORD.

The first word indicates whether the number will be placed at the top or bottom of the page. The second word indicates that the page number is to be printed on every page starting with the first, second, or third page. The third word indicates the position on the page: at the left, center, right, or justified at the outside edge of the page.

The first four sets of six entries have values for each of the six possible subfields. The first set of entries, labeled TYPE N, contain a numerical code for the basic type of subfield increment. A minus one is used to indicate an alphabetic increment, a zero indicates an arabic number in the increment, and a positive number indicates a roman numeral. The positive number indicates the arabic representation of the next roman-numeral page number to be assigned. The second entry set, labeled TYPE A, contains an alphabetic code for the exact type of subfield increment applied. The letter b is used to indicate arabic numbering, c indicates uppercase alphabetic, a indicates lowercase alphabetic, r indicates a lowercase roman subfield, and t is used for uppercase roman subfields. The third set contains the character location for the last character to be incremented in the subfield. The fourth set contains the character location for the last character to be printed in the subfield.

The next two words contain the total number of characters in the page number followed by the actual number of subfields defined by the user. The last item of data in the record is the initialized value of the page number (one character per word). The page number is initialized to zero for arabic numbers and to blanks for all other increment types.

### *Common Storage Structure*

The system stores the page number format in the NUMBRC common area in array PAGE. The initial value for the number is stored in the array INPAGE. A packed page number is stored in array IPAGNO for use by tables and indices.

The first word in PAGE contains the subfield that is to be incremented. The next six words contain the number of characters within the incrementable portion of the subfield. The remaining common area is used as described in the previous section.

### *Related Commands*

.PGNF	.BN	*BN
.DOCU		*PS

### Document Printing Format

#### *Objective*

The system must allow the user to define the basic format for an entire document so that requesting the printing of a document is as simple as possible. The user should have complete control of all data items printed and of the spacing between data items.

The definition of the vertical page format should include the top and bottom margins, header, footer, page number, and the page length. If desired, the system should be able to print the number of the last paragraph printed at the bottom of each page above the standard footer. The user should also be able to place the page number within the header or footer.

The format definition for each paragraph must include the portion of the paragraph number to be printed and all indentions.

The horizontal alignment should include the space between all columns of information and the width and layout of the text within each column.

#### *Data Storage Structure*

The data comprise two portions. The first portion pertains to the page format to be applied. The second portion pertains to the paragraph formats to be applied.

The page format is defined with reference to vertical and horizontal alignment. The column containing the actual text is the only information column for which vertical alignment must be defined. All other data columns are fixed. Several of the items required to define the vertical alignment for the text column are stored separately. The format IDs are the only items required in specifying the formats. The basic vertical page layout is shown in Figure 10. Format IDs for headers, footers, page numbers, and paragraph numbers are given. The total number of lines on one printed page from the top of the page to the very bottom of the page including all margins must be given.

If the last paragraph number printed on the page is to be placed at the bottom of the page, the subfield to be printed must be defined.

The total number of paragraphs forms must also be given.

The horizontal page format is shown in Figure 11. The variable spacings between each column and the total number of characters to be printed across the page must be specified.

Within the text column, additional horizontal alignment is required.

A maximum of six paragraph forms define the horizontal alignment for each paragraph to be printed, as shown in Figure 12.

#### *Data Storage Structure*

All data are stored within the document format table in the form shown in Figure 13.

#### *Common Storage Structure*

All data are located in the /PRINTC/ common in array DFDRC. Data items have been equivalenced for ease of use, as shown in the column definition.

#### *Related Commands*

.DOCU

.FOOT

.HEAD

.PGNF

.PRNF

.PARA

## 9 DEFINING INTERNAL FORMATS FOR PRINTING

### Introduction

The system provides functions for defining the standard format operations such as skipping lines, laying out tables, and copying text. This section describes the functions that the system performs.

### Formatting Tables

The system must provide the typist with a fast method of entering and formatting tables within the text. Tables normally consist of one header that appears as the first text of the table, standard column headers that appear on each page of the table, rows of data separated into columns, and footnotes applied within the text. The type and format of the data which can be accommodated must be easily defined and readily changeable.

### *Data Storage Structure*

Commands are imbedded within the text located in the document Text Table.

### *Common Storage Structure*

The /PRINTC/ common area is used to process all commands.

### *Related Commands*

*TB*	*R*	*TE*
*TH*	*N*	

## Blocking Text

### *Objective*

The system must allow the user to request that a particular segment of text be printed on the same page and not divided onto two pages.

When the text is longer than one printed page, printing should start on a new page.

### *Data Storage Structure*

Commands are imbedded within the text located in the document Text Table.

### *Common Storage Structure*

The /PRINTC/ common area is used to process all commands.

### *Related Commands*

\*B\*      .BL.

\*BE\*     .UB.

## Justifying Text

### *Objective*

The system must provide functions to left-, center-, and right-justify text to any column specified from either the left or right page margin.

### *Data Storage Structure*

Commands are imbedded within the text located in the document Text Table.

### *Common Storage Structure*

The /PRINTC/ common area is used to process all commands.

### *Related Commands*

\*CJ\*

\*LJ\*

\*RJ\*

### Underlining Text

#### *Objective*

The system must permit underlining of text on both high- and low-speed printers.

### *Data Storage Structure*

Commands are imbedded within the text located in the document Text Table.

### *Common Storage Structure*

The /PRINTC/ common area is used to process all commands.

### *Related Commands*

\*U\*

\*UE\*

### Skipping Pages and Lines

When desired, the user can instruct the system to skip to the next clean page or clean line. The user can also skip several blank pages or lines.

#### *Data Storage Structure*

Commands are imbedded within the text located in the document Text Table.

#### *Common Storage Structure*

The /PRINTC/ common area is used to process all commands.

#### *Related Commands*

.SP

.SL

#### Footnotes

#### *Objective*

The system should place footnotes at the bottom of each page or at the end of each document.

#### *Data Storage Structure*

Commands are imbedded within the text located in the Text Table.

#### *Common Storage Structure*

The /PRINTC/ common area is used to process all commands.

#### *Related Commands*

\*FD\*      \*FE\*

\*FP\*

## Compiling a Table of Contents, List of Figures, and Index

### *Objective*

The system must create a table of contents and various types of lists that are ordered by page number. It should also create indices in alphabetical order.

### *Data Storage Structure*

Commands are imbedded within the text located in the document Text Table.

### *Common Storage Structure*

The /PRINTC/ common area is used to process all commands.

### *Related Commands*

*IX*	.IX	.PI
*TC*	.TC	.PT

## 10 PRINTING INFORMATION

### Introduction

Once the basic formats for the data to be printed have been defined, a print command can be issued. The EDITSPEC system provides two basic types of print commands. The first will print a complete document or a portion of it; the second will print the information for all documents logically combined into one project.

### Printing a Document or Portion Thereof

#### *Objective*

When a complete document or a portion thereof is to be printed, the document printing format, the column information to be printed, and the spacing of the function must be under the user's control. The user must be able to instruct the system to reproduce any page that has been previously printed. The system should produce tables of contents, lists, and indices as required during printing. Before printing begins, the system should check to insure that the text to be printed is complete. The user should be able to route the output to the central site printer, the office printer, or to his own terminal. Also, the user should be able to request that the computer check the spacing between words and sentences to insure that no spacing errors occur.

#### *Data Storage Structure*

All data storage structures have been defined previously in the format section.

### *Common Storage Structure*

The storage required in printing a document is divided into four basic common areas.

`/NUMBRC/` common contains all information related to the page and paragraph numbers which applies to printing the document. See the earlier sections on page numbering format and paragraph numbering format (Chapter 9) for detailed definitions and descriptions of the data structures.

`/PRINTC/` common contains the general information that is passed from one subroutine to another in performing work.

The `/PAGEBF/` common area contains all formatted information for the next page to be printed. Five types of column information can be printed from left to right in the order described herein. The ID of the first flag in the formatted line is given in the `FLAG` array. If the formatted line contains more than one flag ID, the first ID is preceded by a negative sign.

The text table line number for the last character printed on the formatted line is located in the `LNNUM` array. If the last character on the line came from another document through a "copy no move" command, the line number of the other document will be given, preceded by a negative sign.

The body of the text is located in the `PGBUF` array. Since FORTRAN stores characters in a column major system, the columns of this array correspond to the lines on the printed page. The rows then correspond to the text of the line. The system will allow a maximum of 61 printable lines. The structure of `PGBUF` is shown in Figure 14.

The name of the document from which the last character on the printed line was copied is placed in the AUDDOC array. If the text was entered directly into this document by a user, the name is left blank. The audit cycle number for the document copied from and the current document cycle numbers are stored within the AUDTCN array.

The text segment assigned to the last character on the line is stored in the TEXTSG array.

/FORMAT/ common contains the definitions of all variable printing format statements.

#### *Related Commands*

.PR

.BN

\*BN\*

.BP

\*BP\*

### Printing a Complete Project Specification

#### *Objective*

The system must provide a simple method for printing one complete project specification.

#### *Storage Structures*

... The data storage structure and the common storage structure are the same as for printing a document. The system will print the documents in the order given in the MASTERSP document for the project.

#### *Related Commands*

.PPRO

## 11 COMMAND EXECUTION

### Objective

The system should allow the user to control when the commands are executed. In some instances, the user must make exactly the same changes to several documents. In such cases the user should be required to enter the changes into the system only once. The user should not have to duplicate his efforts at any time. For this reason, the system should support both the immediate execution and <sup>deferred</sup>~~deferred~~ execution of commands.

### Data Storage Structure

Those commands that are executed immediately are not stored. <sup>e</sup>Differed commands are entered as text into the Text Table.

### Common Storage Structure

No common area is required.

### Related Commands

.EXEC

.EX

## 12 DATA BASE PROTECTION

The system provides a periodic backup for all system tables, documents, and datasets.

The backup and restoration programs are described in detail in the operations and maintenance manual and are not discussed here.

### 13 PROGRAM AND DATA CONVERSION

The system is designed and implemented to provide economical program conversion as well as economical production. The system is also designed to provide for economical data conversion from one machine to another. The data stored are as machine independent as possible.

The data and program conversion processes are described in detail in EDITSPEC Systems Manual No. 5 and hence are not discussed here.

# USER TABLE

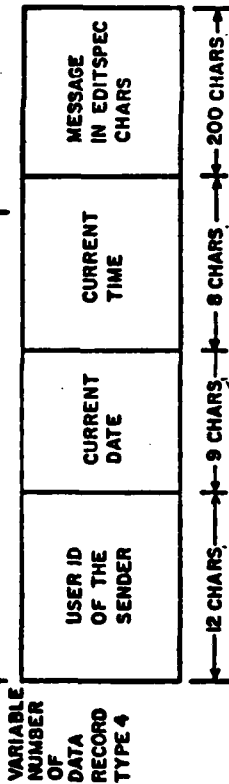
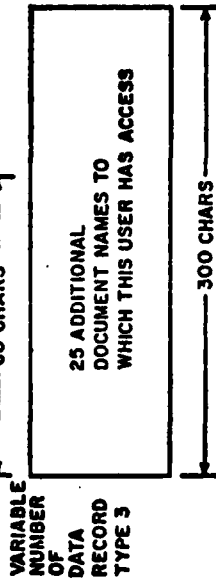
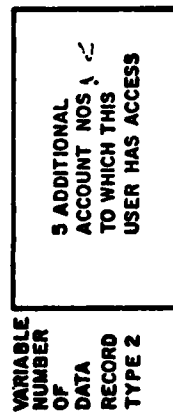
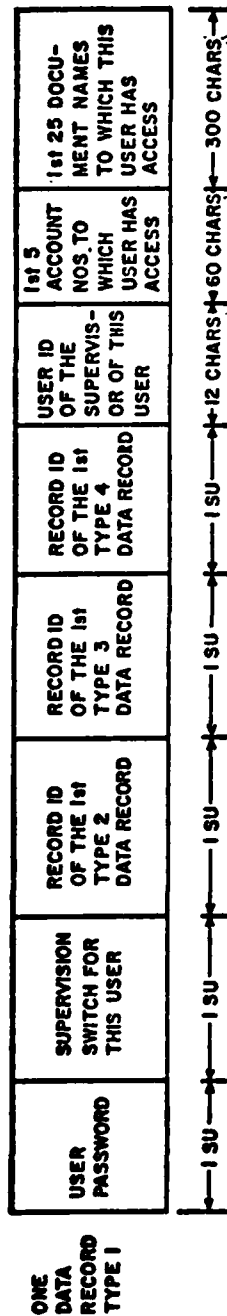


FIGURE 1 USER INFORMATION STRUCTURE

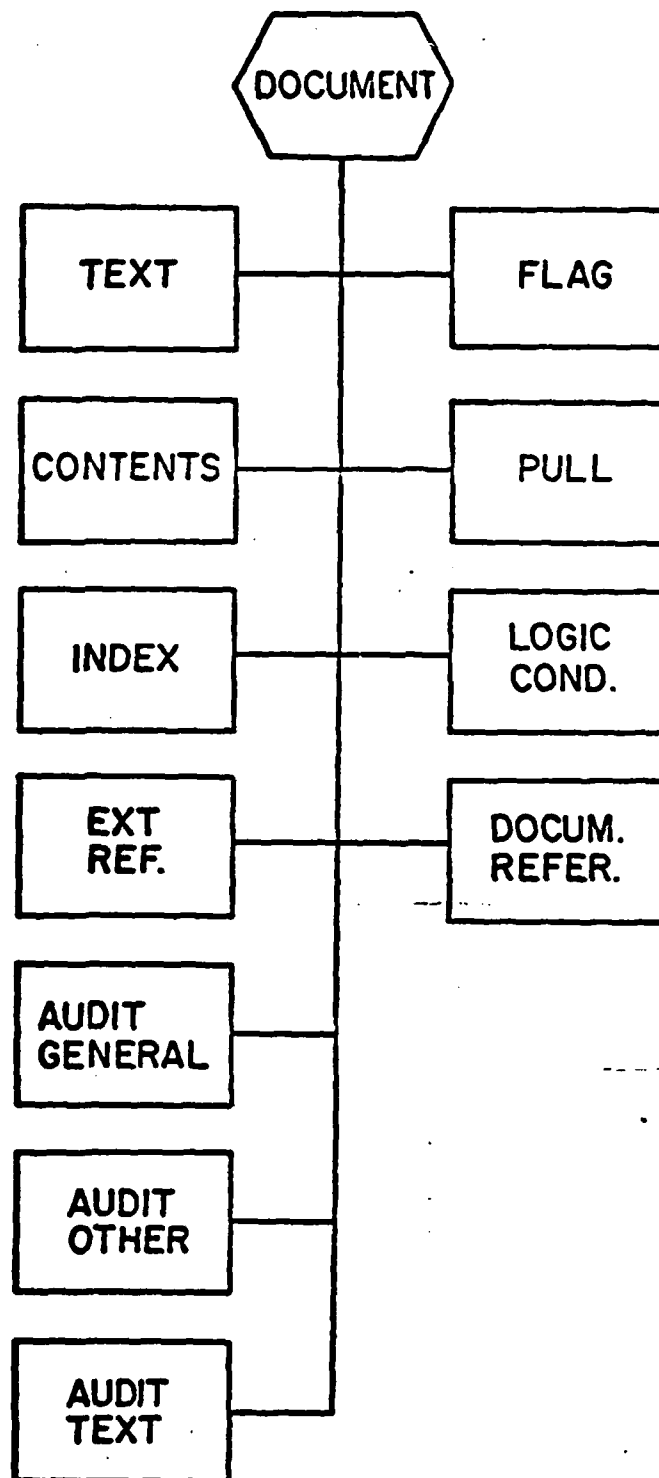


FIGURE 2 DOCUMENT TABLE STRUCTURE

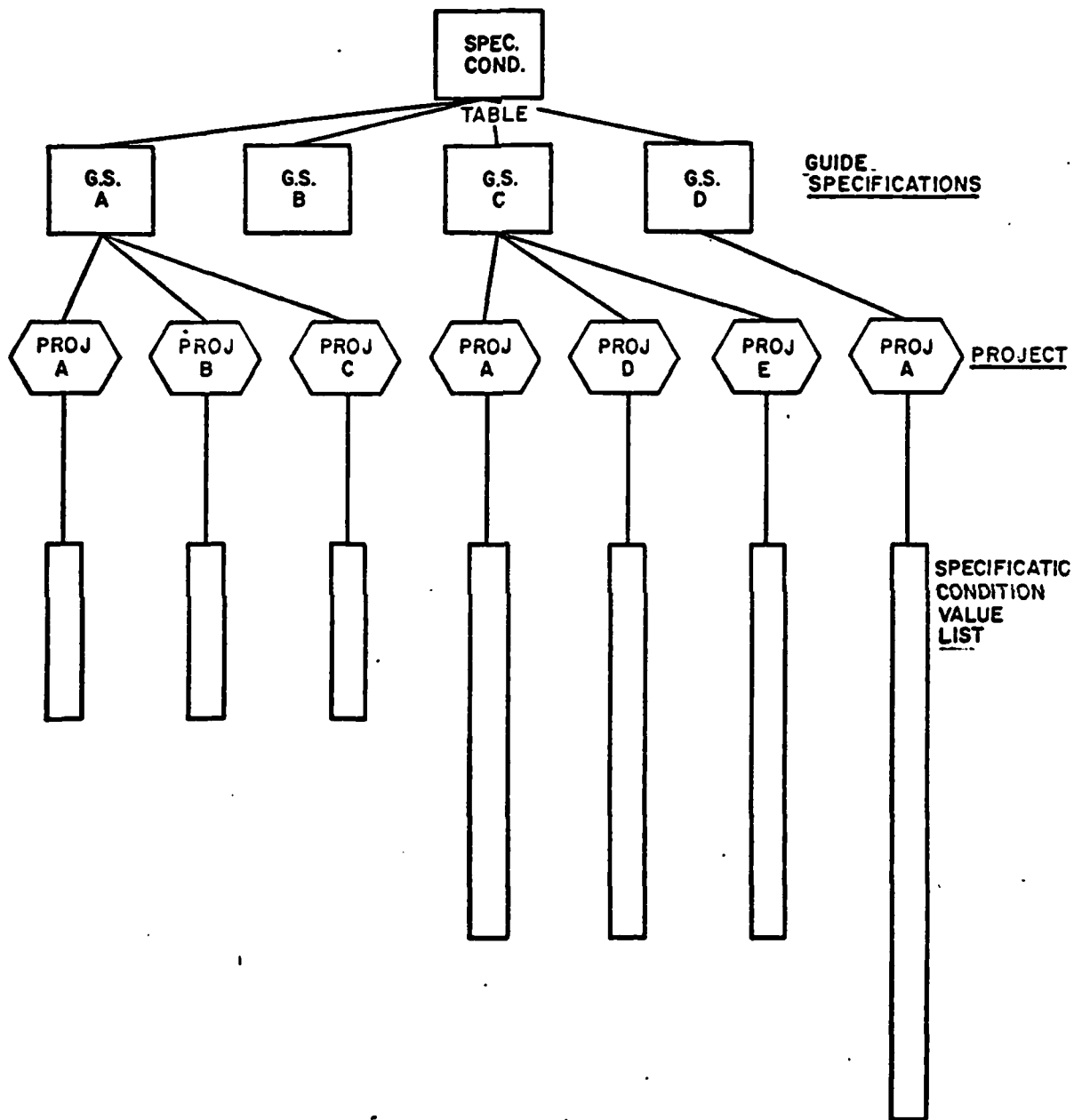
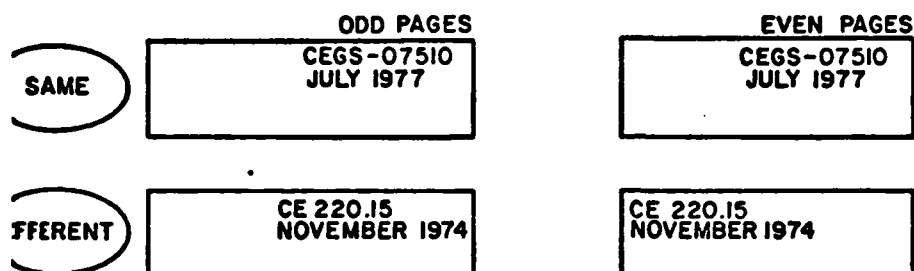


FIGURE 4 SPECIFICATION CONDITION STRUCTURE

# HEADERS

FAMILY HOUSING  
JULY 1978  
(SECTION TITLE)



## DATA STORAGE STRUCTURE

KEY		LENGTHS			
FORMAT ID	ODD	EVEN	ODD (STANDARD)	EVEN	

## COMMON STORAGE STRUCTURE

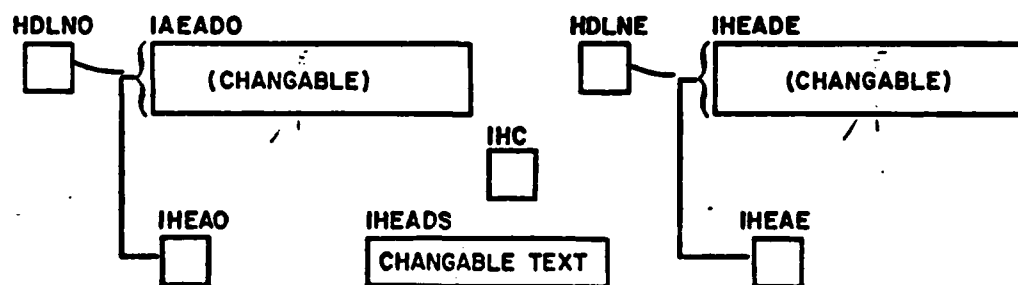


FIGURE 5 HEADER DATA STRUCTURE

NOTES

FAMILY HOUSING  
JULY 1978  
(SECTION TITLE)

X  
SAME

ODD PAGES

CEGS-07510  
JULY 1977

EVEN PAGES

CEGS-07510  
JULY 1977

X  
DIFFERENT

CE 220.15  
NOVEMBER 1974

CE 220.15  
NOVEMBER 1974

DATA  
STORAGE  
STRUCTURE

KEY		LENGTHS			
FORMAT	ID	ODD	EVEN	ODD (STANDARD)	EVEN

COMMON  
STORAGE  
STRUCTURE

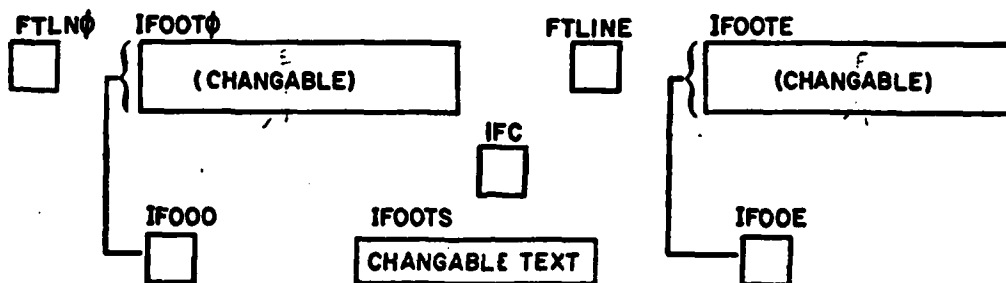
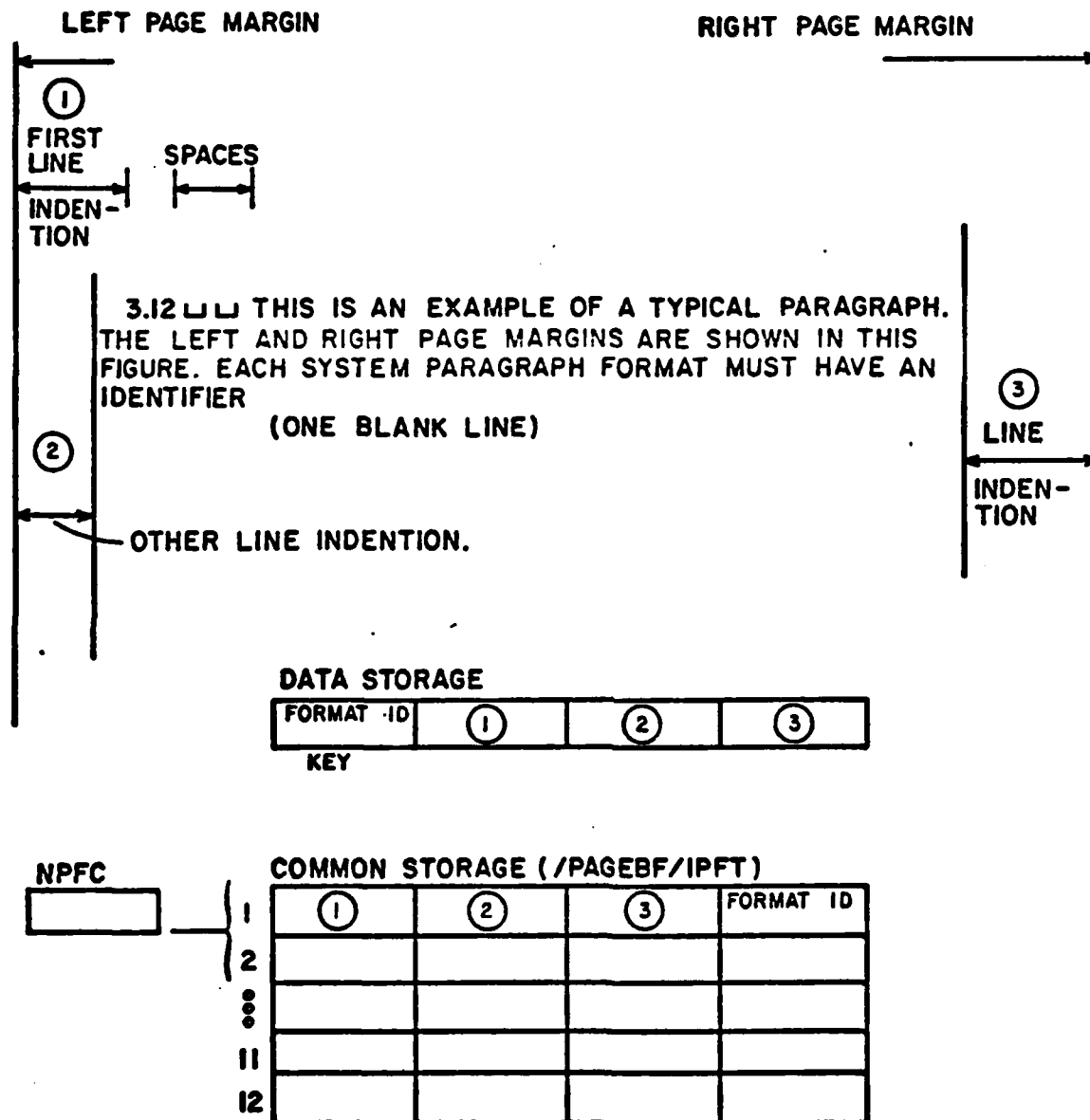


FIGURE 6 FOOTER DATA STRUCTURE



**FIGURE 7. TYPICAL PARAGRAPH FORMAT AND DATA STRUCTURE**

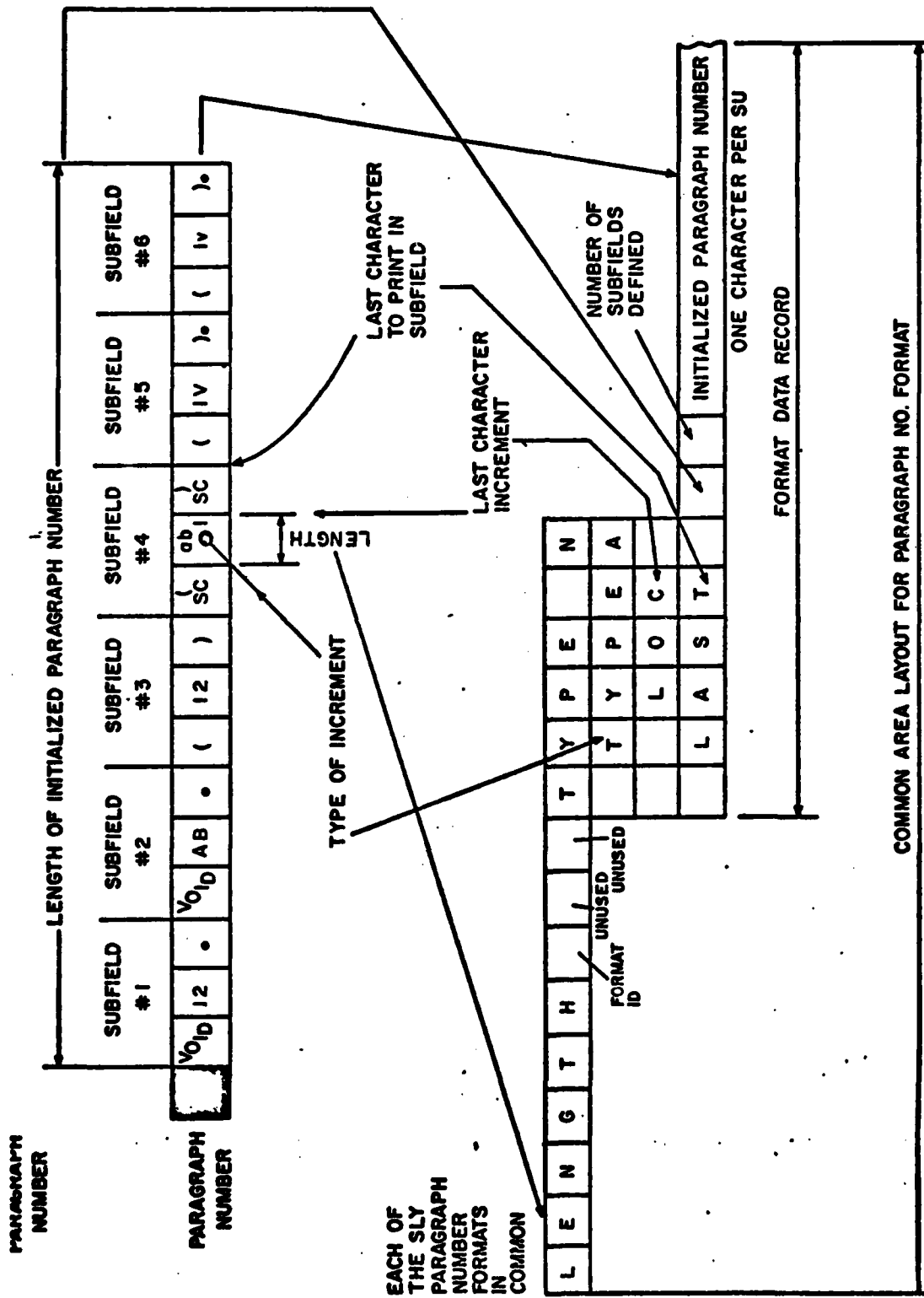
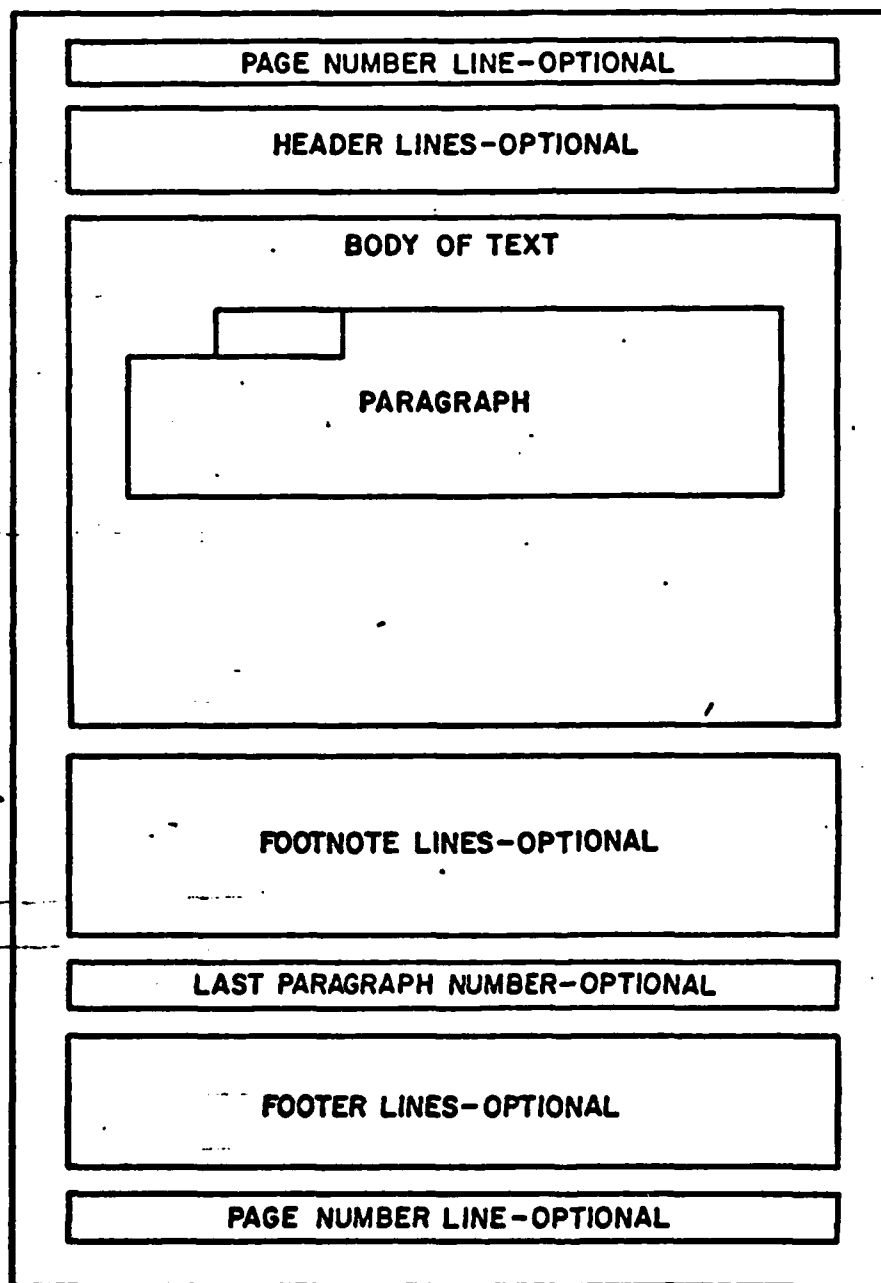


FIGURE 8 SYSTEM DESIGN -- PARAGRAPH NUMBERING FORMAT



**FIGURE 10 VERTICAL PAGE LAYOUT**

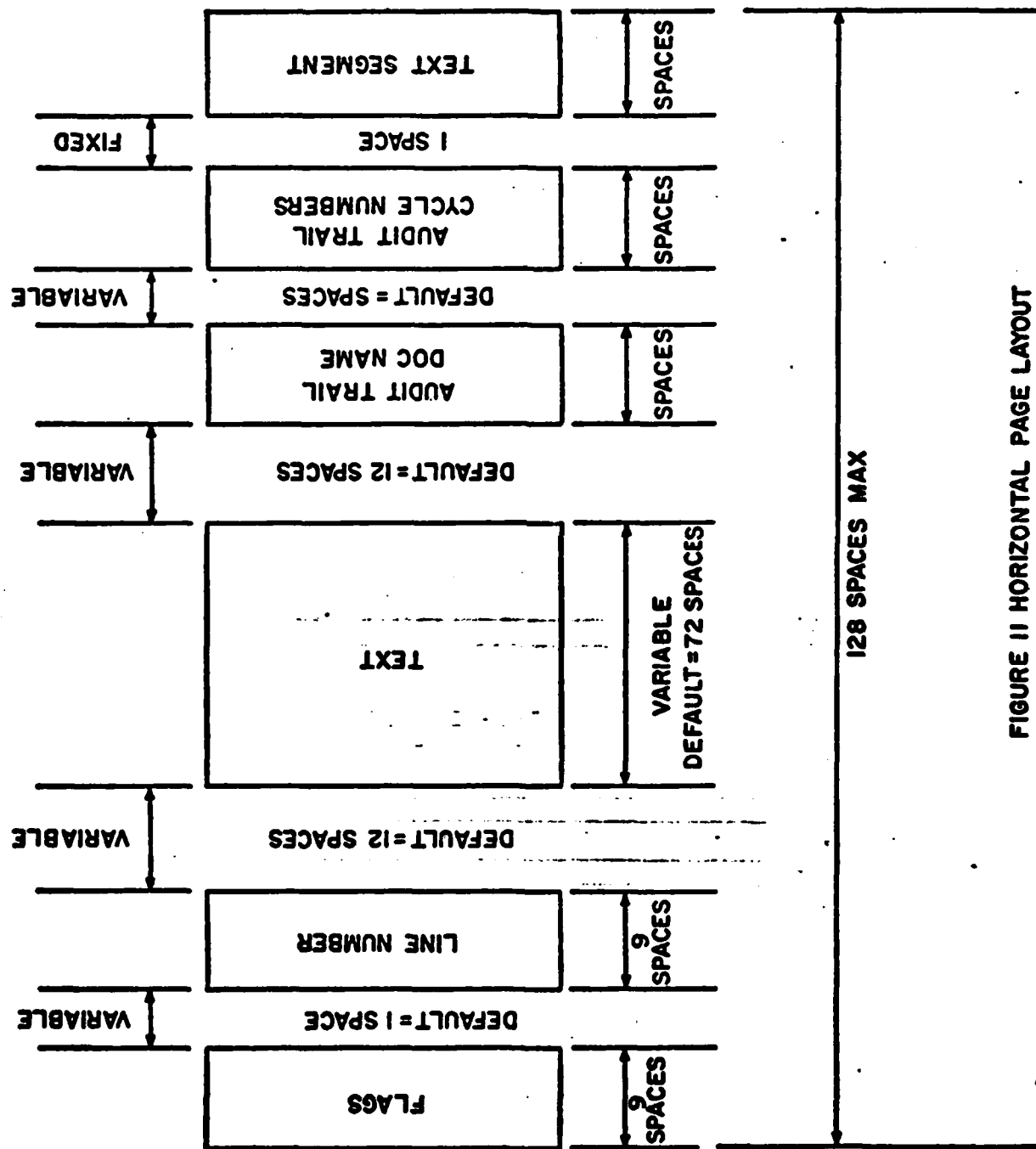


FIGURE 11 HORIZONTAL PAGE LAYOUT

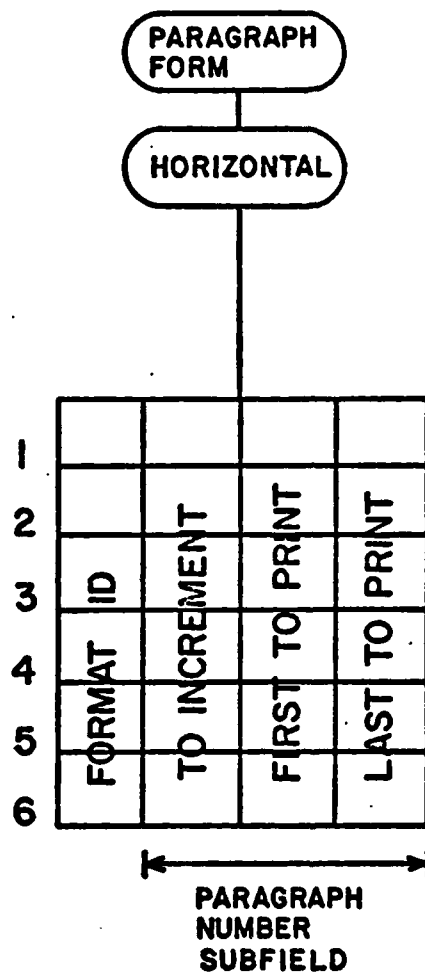


FIGURE 12 PARAGRAPH FORM STRUCTURE

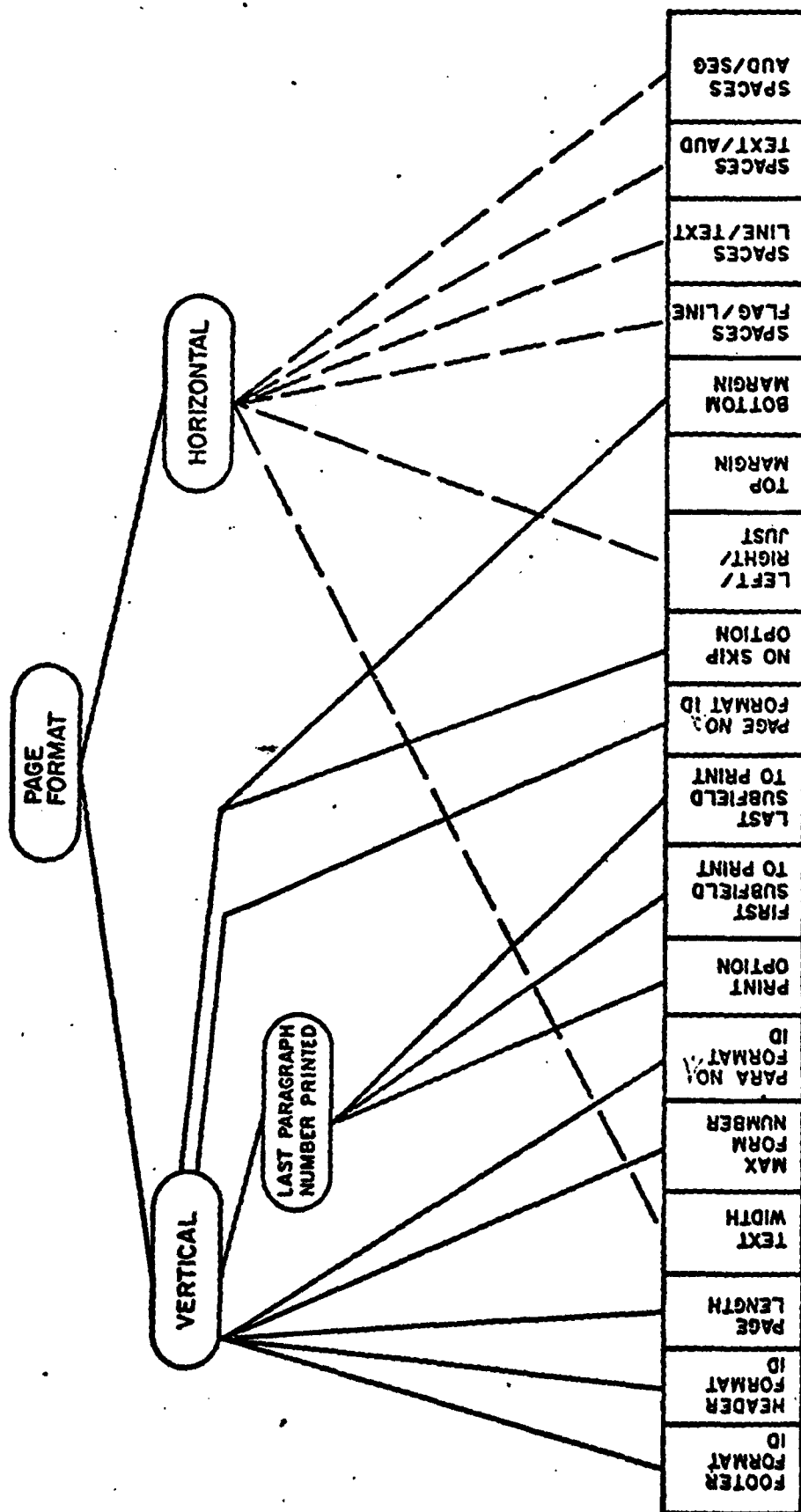


FIGURE 13 PAGE FORMAT STRUCTURE

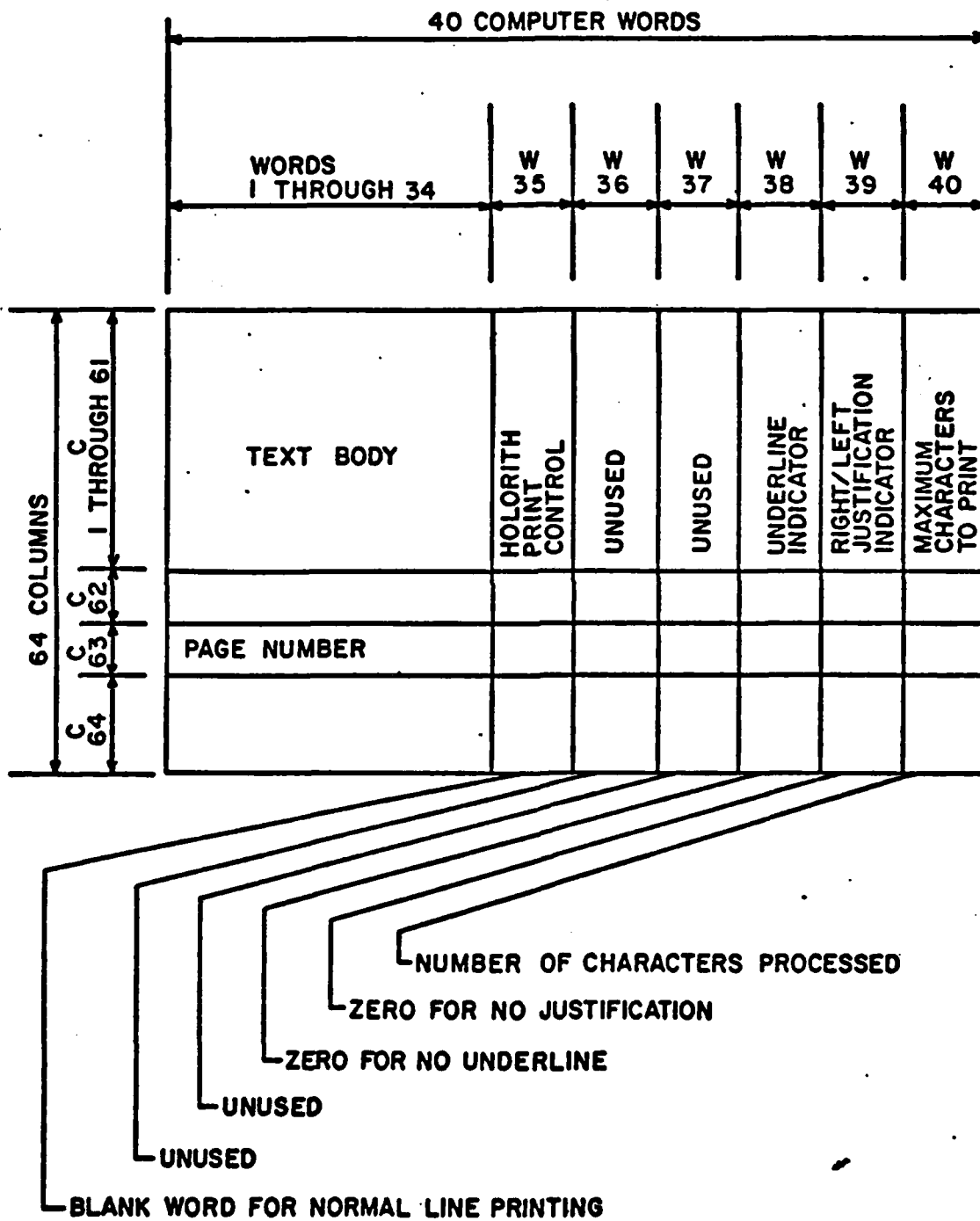


FIGURE 14 PGBUF STRUCTURE

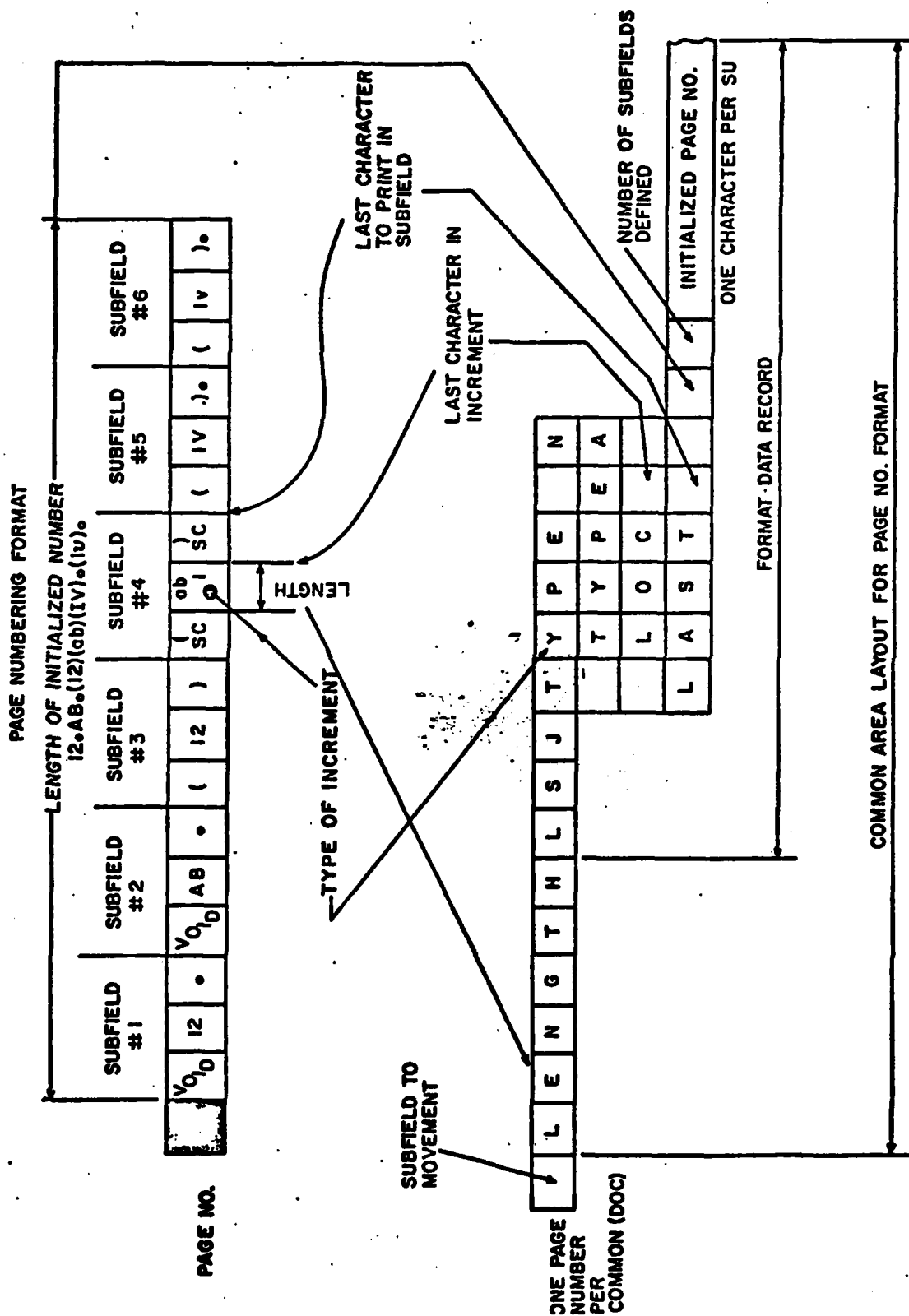


FIGURE 9 SYSTEM DESIGN — PAGE NUMBERING FORMAT.

**END**

**FILMED**

**3-83**

**DTIC**